Optimal Stopping via Randomized Neural Networks

Calypso Herrera[1] Florian Krach[1] Pierre Ruyssen[2]
Josef Teichmann[1]

June 22th, 2021

[1]Department of Mathematics, ETH Zurich
[2]Google Brain, Google Zurich

Optimal stopping problem?

Exploration-exploitation of **standard approaches** and **machine learning based approaches** to solve the optimal stopping problem.

Least Squares Monte Carlo

- The price of the discretized American option can be expressed through the Snell envelope U.
- underlying process X, $x^i = (x_0, x_1^i, \dots, x_N^i)$ for path $i \in \{1, \dots m\}$
- cash flow $p^i = (p_0, p_1^i, \dots, p_N^i)$ for path $i \in \{1, \dots m\}$
- α the step-wise discounting factor
- continuation value $c_n(x) = E(\alpha U_{k+1} | X_n = x)$
- **Tsitsiklis and Van Roy (2001)**

$$
\begin{aligned}
{p_N}^i &= g({x_N}^i) \\
{p_n}^i &= \max\Big(\underbrace{g({x_n}^i)}_{\text{stop}},\ \underbrace{c_n({x_n}^i)}_{\text{continue}}\Big)
\end{aligned}
$$

Least Squares Monte Carlo

Tsitsiklis and Van Roy 2001

$$
\begin{aligned}
p_N^i &= g(x_N^i) \\
p_n^i &= \underbrace{g(x_n^i)}_{\text{payoff}} \underbrace{1_{\{g(x_n^i) \geq c_n(x_n^i)\}}}_{\text{exercise}} + \underbrace{c_n(x_n^i)}_{\text{continuation value}} \underbrace{1_{\{g(x_n^i) < c_n(x_n^i)\}}}_{\text{continue}}
\end{aligned}
$$

- **Longstaff and Schwartz (2001)** use the approximation only for the stopping decision.

Longstaff and Schwartz 2001

$$
\begin{aligned}
p_N^i &= g(x_N^i) \\
p_n^i &= \underbrace{g(x_n^i)}_{\text{payoff}} \underbrace{1_{\{g(x_n^i) \geq c_n(x_n^i)\}}}_{\text{exercise}} + \underbrace{\alpha p_{n+1}^i}_{\text{discounted future price}} \underbrace{1_{\{g(x_n^i) < c_n(x_n^i)\}}}_{\text{continue}}
\end{aligned}
$$

Linear combination of features for the continuation value

Longstaff and Schwartz 2001

$$
\begin{aligned}
p_N^i &= g(x_N^i) \\
p_n^i &= \underbrace{g(x_n^i)}_{\text{payoff}} \underbrace{1_{\{g(x_n^i) \geq c_n(x_n^i)\}}}_{\text{exercise}} + \underbrace{\alpha p_{n+1}^i}_{\text{discounted future price}} \underbrace{1_{\{g(x_n^i) < c_n(x_n^i)\}}}_{\text{continue}}
\end{aligned}
$$

- The continuation value is approximated by linear combination of basis functions,

$$
c_n(x) = E(\alpha U_{n+1} | x_n = x) \approx \sum_{i=1}^{n} \theta_i f_i(x) = \theta^\top f(x)
$$

- where the parameters θ are found by minimizing the loss function

$$
\varphi(\theta_n) = \sum_{i=1}^{N} \left(c_{\theta_n}(x_n^i) - \alpha p_{n+1}^i \right)^2
$$

- using a linear regression **at each date** $n \in \{N-1, \ldots, 1\}$.

Neural networks approximation of the continuation value

Kohler et. al. (2010) = Tsitsiklis Van Roy(2001) + neural networks

$$\begin{aligned} p_N^i &= g(x_N^i) \\ p_n^i &= \underbrace{g(x_n^i)}_{\text{payoff}} \underbrace{1_{\{g(x_n^i) \geq c_{\theta_n}(x_n^i)\}}}_{\text{exercise}} + \underbrace{c_{\theta_n}(x_n^i)}_{\text{continuation value}} \underbrace{1_{\{g(x_n^i) < c_{\theta_n}(x_n^i)\}}}_{\text{continue}} \end{aligned}$$

- The continuation value is now approximated by a neural network

$$c_k(x) \approx c_{\theta_n}(x)$$

- where the parameters θ_n are found by minimizing the loss function

$$\varphi(\theta_n) = \sum_{i=1}^{N} \left(c_{\theta_n}(x_n^i) - \alpha p_{n+1}^i \right)^2 .$$

- using a gradient descent method at **each date** $n \in \{N-1, \ldots, 1\}$.

Neural networks approximation of the continuation value

Lapeyre Lelong (2019) = Longstaff Schwartz (2001) + neural networks

$$\begin{aligned} p_N^i &= g(x_N^i) \\ p_n^i &= \underbrace{g(x_n^i)}_{\text{payoff}} \underbrace{1_{\{g(x_n^i) \geq c_{\theta_n}(x_n^i)\}}}_{\text{exercise}} + \underbrace{\alpha p_{n+1}^i}_{\text{discounted future price}} \underbrace{1_{\{g(x_n^i) < c_{\theta_n}(x_n^i)\}}}_{\text{continue}} \end{aligned}$$

- The continuation value is now approximated by a neural network

$$c_k(x) \approx c_{\theta_n}(x)$$

- where the parameters θ_n are found by minimizing the loss function

$$\varphi(\theta_n) = \sum_{i=1}^{N} \left(c_{\theta_n}(x_n^i) - \alpha p_{n+1}^i \right)^2 .$$

- using a gradient descent method at **each time** $n \in \{N-1, \dots, 1\}$.

Neural networks approximation of the indicator function

Becker et. al. (2019) = Longstaff Schwartz (2001) + NN for indicator

$$\begin{aligned} p_N^i &= g(x_N^i) \\ p_n^i &= \underbrace{g(x_n^i)}_{\text{payoff}} \underbrace{1_{\{g(x_n^i) \geq c_k(x_n^i)\}}}_{f_{\theta_n}(x)} + \underbrace{\alpha p_{n+1}^i}_{\text{discounted future price}} \underbrace{1_{\{g(x_n^i) < c_{\theta_n}(x_n^i)\}}}_{1 - f_{\theta_n}(x)} \end{aligned}$$

- The indicator function is now approximated by a neural network

$$1_{\{g(x_n^i) \geq c_k(x_n^i)\}} \approx f_{\theta_n}(x)$$

- where the parameters θ_n are found by ~~minimizing~~ maximizing the loss function

$$\varphi(\theta_n) = \sum_{i=1}^{N} \alpha p_k^i \, .$$

- using a gradient descent method at **each time** $n \in \{N-1, \ldots, 1\}$.

Randomized Least Squares Monte Carlo (RLSM)

Randomized Least Squares Monte Carlo (RLSM)

$$\begin{aligned} p_N^i &= g(x_N^i) \\ p_n^i &= \underbrace{g(x_n^i)}_{\text{payoff}} \underbrace{\mathbf{1}_{\{g(x_n^i) \geq c_\theta(x_n^i)\}}}_{\text{exercise}} + \underbrace{\alpha p_{n+1}^i}_{\text{discounted future price}} \underbrace{\mathbf{1}_{\{g(x_n^i) < c_\theta(x_n^i)\}}}_{\text{continue}} \end{aligned}$$

- For each date n, going backward, the continuation value is approximated by a neural network

$$c_{\theta_n}(x) = A_n^\top \sigma(Ax + b) + b_n$$

- where the parameters of the hidden layer (A,b) are randomly chosen and not optimized
- and only the parameters of the last layer $\theta_n = (A_n, b_n)$ are optimized by minimizing the loss function

$$\psi_n(\theta_n) = \sum_{i=1}^{m} \left(c_{\theta_n}(x_n^i) - \alpha p_{n+1}^i \right)^2 . \tag{1}$$

Randomized Least Squares Monte Carlo (RLSM)

Algorithm 1 Optimal stopping via randomized least squares Monte Carlo (RLSM)

Input: discount factor α, initial value x_0
Output: price p_0
1: sample a random matrix $A \in \mathbb{R}^{(K-1)\times d}$ and a random vector $b \in \mathbb{R}^{K-1}$
2: simulate $2m$ paths of the underlying process $(x_1^i, \dots, x_N^i)$ for $i \in \{1, \dots, 2m\}$
3: for each path $i \in \{1, \dots, 2m\}$, set $p_N^i = g(x_N^i)$
4: for each date $n \in \{N-1, \dots, 1\}$
  a: for each path $i \in \{1, \dots, 2m\}$, set $\phi(x_n^i) = (\sigma(Ax_n^i + b)^\top, 1)^\top \in \mathbb{R}^K$
  b: set $\theta_n = \alpha \left(\sum_{i=1}^m \phi(x_n^i)\phi^\top(x_n^i)\right)^{-1} \left(\sum_{i=1}^m \phi(x_n^i) p_{n+1}^i\right)$
  c: for each path $i \in \{1, \dots, 2m\}$, set $p_n^i = g(x_n^i)\mathbf{1}_{g(x_n^i)\geq\theta_n^\top\phi(x_n^i)} + \alpha p_{n+1}\mathbf{1}_{g(x_n^i)<\theta_n^\top\phi(x_n^i)}$
5: set $p_0 = \max(g(x_0), \frac{1}{m}\sum_{i=m+1}^{2m} \alpha p_1^i)$

Theorem (informal)

As the number of sampled paths m and the number of random basis functions K go to ∞, the price p_0 computed with Algorithm 1 converges to the correct price of the Bermudan option.

An alternative to the backward recursion

- Instead of having a different function approximator for each date, **we use the same function approximator for all dates**

$$C_n(x_n) \approx C_\theta(n, x_n) \quad \forall n \in \{1, \dots, N-1\}$$

- The algorithm is initialized with a parameter vector θ_0, and **after one sample path the parameter vector is updated** following

$$\theta_{n+1} = \theta_n + \alpha \nabla \varphi(\theta)$$

- This is the principle of **Reinforcement Learning**.
- We first make some random optimal stopping decisions and then we reinforce our learning over the iterations.
- **This is not new!** The first who introduce the reinforcement learning for the optimal stopping were **Tsitsiklis and Van Roy (1999)**.

Fitted Q-learning (FQI)

Tsitsiklis Van Roy(1999) = Off-Policy Q-learning + linear approximator

$$\begin{aligned} p_N^i &= g(x_N^i) \\ p_n^i &= \underbrace{g(x_n^i)}_{\text{Payoff}} \underbrace{1_{\{g(x_n^i) \geq c_\theta(x_n^i)\}}}_{\text{exercise}} + \underbrace{c_\theta(x_n^i)}_{\text{continuation value}} \underbrace{1_{\{g(x_n^i) < c_\theta(x_n^i)\}}}_{\text{continue}} \end{aligned}$$

- The continuation value is approximated by a linear combination of basis functions $C(n, x_n) \approx \theta^T f(n, x)$
- Where the parameters θ are found by minimizing the loss

$$\varphi(\theta) = \sum_{i=1}^{m} \sum_{n=1}^{N} \left(c_\theta(n, x_n^i) - \alpha p_{n+1}^i\right)^2$$

- using **only one** linear regression for all dates.

Randomized Fitted Q-learning (RFQI)

$$\begin{aligned} p_N^i &= g(x_N^i) \\ p_n^i &= \underbrace{g(x_n^i)}_{\text{payoff}} \underbrace{\mathbf{1}_{\{g(x_n^i)\geq c_\theta(x_n^i)\}}}_{\text{exercise}} + \underbrace{c_\theta(x_n^i)}_{\text{continuation value}} \underbrace{\mathbf{1}_{\{g(x_n^i)< c_\theta(x_n^i)\}}}_{\text{continue}} \end{aligned}$$

- As for reinforcement learning algorithms, we first fix the parameters θ^0 and then we update them $\theta^1, \theta^2, \ldots$ through multiples iterations.
- For all dates n, the continuation value is approximated by a neural network

$$c_\theta(\tilde{x}) = A_2^\top \sigma(A\tilde{x} + b) + b_2$$

- where $\tilde{x}_n = (n, x_n)$.
- where the parameters of the hidden layer (A,b) are randomly chosen and not optimized
- and only the parameters of the last layer $\theta = (A_2, b_2)$ are optimized by minimizing the loss function

$$\psi(\theta) = \sum_{n=1}^{N} \sum_{i=1}^{m} \left(c_{\theta_n}(\tilde{x}_n^i) - \alpha p_{n+1}^i \right)^2 . \tag{2}$$

Algorithm 2 Optimal stopping via randomized fitted Q-Iteration (RFQI)

Input: discount factor α, initial value x_0
Output: price p_0
1: sample a random matrix $A \in \mathbb{R}^{(K-1)\times(d+2)}$ and a random vector $b \in \mathbb{R}^{K-1}$
2: simulate $2m$ paths of the underlying process $(x_1^i, \dots, x_N^i)$ for $i \in \{1, \dots, 2m\}$
3: initialize weights $\theta = 0 \in \mathbb{R}^K$
4: for each iteration ℓ until convergence of θ
  a: for each path $i \in \{1, \dots, 2m\}$,
    i: set $p_N^i = g(x_N^i)$
    ii: for each date $n \in \{1, \dots, N-1\}$,
      set $\phi(n, x_n^i) = (\sigma(A\tilde{x}_n^i + b), 1) \in \mathbb{R}^K$
      set $p_n^i = \max(g(x_n^i), \phi(n, x_n^i)^\top \theta)$
  b: set $\theta = \alpha \left(\sum_{n=1}^N \sum_{i=1}^m \phi(n, x_n^i)\phi^\top(n, x_n^i)\right)^{-1} \cdot \left(\sum_{n=1}^N \sum_{i=1}^m \phi(n, x_n^i) p_{n+1}^i\right) \in \mathbb{R}^K$
5: set $p_0 = \max(g(x_0), \frac{1}{m} \sum_{i=m+1}^{2m} \alpha p_1^i)$

Theorem (informal)

As the number of iterations L, the number of sampled paths m and the number of random basis functions K go to ∞, the price p_0 computed with Algorithm 2 converges to the correct price of the Bermudan option.

		price						duration					
d	x_0	LSM	DOS	NLSM	**RLSM**	FQI	**RFQI**	LSM	DOS	NLSM	**RLSM**	FQI	**RFQI**
5	80	5.80 (0.06)	5.96 (0.05)	5.86 (0.13)	**5.64 (0.05)**	6.08 (0.13)	**6.07 (0.11)**	15s	22s	9s	**6s**	8s	**6s**
	100	26.15 (0.16)	26.82 (0.20)	26.35 (0.19)	**25.36 (0.16)**	26.78 (0.14)	**26.68 (0.25)**	15s	23s	13s	**6s**	8s	**6s**
	120	50.87 (0.13)	51.50 (0.25)	51.00 (0.17)	**49.89 (0.29)**	51.70 (0.20)	**51.67 (0.17)**	15s	23s	13s	**6s**	8s	**6s**
10	80	9.88 (0.09)	10.29 (0.12)	9.93 (0.19)	**9.63 (0.07)**	10.40 (0.12)	**10.36 (0.10)**	30s	23s	9s	**6s**	12s	**7s**
	100	35.12 (0.11)	35.82 (0.14)	35.10 (0.25)	**34.18 (0.15)**	35.92 (0.13)	**35.92 (0.11)**	30s	23s	12s	**6s**	12s	**7s**
	120	61.63 (0.22)	62.64 (0.21)	61.75 (0.21)	**60.67 (0.16)**	62.72 (0.22)	**62.77 (0.13)**	30s	23s	12s	**6s**	14s	**7s**
50	80	23.03 (0.09)	24.46 (0.11)	22.72 (0.31)	**22.85 (0.07)**	24.59 (0.13)	**24.47 (0.07)**	7m59s	27s	12s	**7s**	5m25s	**8s**
	100	53.56 (0.09)	55.00 (0.12)	52.45 (0.41)	**53.13 (0.11)**	55.34 (0.12)	**55.08 (0.14)**	7m58s	26s	13s	**7s**	6m 9s	**8s**
	120	83.90 (0.20)	85.76 (0.12)	83.12 (0.72)	**83.36 (0.15)**	85.88 (0.13)	**85.79 (0.20)**	8m 1s	26s	13s	**7s**	6m22s	**8s**
100	80	26.16 (0.09)	30.47 (0.16)	27.64 (0.55)	**29.13 (0.10)**	30.29 (0.09)	**30.58 (0.12)**	35m39s	31s	13s	**8s**	1h17m59s	**9s**
	100	57.89 (0.14)	62.64 (0.25)	59.04 (0.72)	**60.93 (0.10)**	62.27 (0.21)	**62.79 (0.19)**	36m59s	31s	14s	**8s**	1h18m15s	**9s**
	120	89.00 (0.15)	94.64 (0.22)	90.20 (1.19)	**92.65 (0.04)**	94.35 (0.14)	**94.91 (0.15)**	37m 0s	31s	14s	**8s**	1h16m15s	**9s**
500	80	-	42.85 (0.14)	39.18 (1.12)	**42.90 (0.05)**	-	**44.03 (0.08)**	-	1m25s	27s	**14s**	-	**15s**
	100	-	78.10 (0.14)	73.59 (1.11)	**78.22 (0.09)**	-	**79.50 (0.12)**	-	1m41s	30s	**14s**	-	**15s**
	120	-	113.27 (0.26)	107.55 (1.26)	**113.37 (0.12)**	-	**115.18 (0.17)**	-	1m44s	31s	**14s**	-	**15s**
1000	80	-	47.91 (0.12)	44.85 (1.07)	**48.56 (0.07)**	-	**49.66 (0.04)**	-	2m58s	45s	**22s**	-	**23s**
	100	-	84.43 (0.15)	80.68 (1.20)	**85.26 (0.08)**	-	**86.56 (0.11)**	-	2m59s	42s	**22s**	-	**23s**
	120	-	121.03 (0.12)	116.82 (1.07)	**121.88 (0.15)**	-	**123.36 (0.21)**	-	2m58s	42s	**22s**	-	**22s**
2000	80	-	51.18 (0.15)	50.71 (0.62)	**54.10 (0.08)**	-	**55.06 (0.07)**	-	6m18s	1m10s	**38s**	-	**37s**
	100	-	88.53 (0.13)	87.82 (0.79)	**92.19 (0.07)**	-	**93.29 (0.10)**	-	6m17s	1m14s	**39s**	-	**38s**
	120	-	125.88 (0.21)	125.24 (0.84)	**130.21 (0.12)**	-	**131.53 (0.07)**	-	6m 7s	1m11s	**38s**	-	**37s**

Table: Max call option on Black Scholes for different number of stocks d and varying initial stock price x_0. RFQI achieves the highest prices while being the fastest and having considerably less trainable parameters.

LSM: Least Squares Monte Carlo (Longstaff and Schwartz, 2001)
DOS: Deep Optimal Stopping (Becker et al., 2019)
NLSM: Neural Least Squares Monte Carlo (Lapeyre and Lelong, 2019)
RLSM: Randomized Least Squares Monte Carlo
FQI: Fitted Q-Iteration (Tsitsiklis and Van Roy, 2001)
RFQI: Randomized Fitted Q-Iteration

Heston - Max Call

	price						duration					
d	LSM	DOS	NLSM	**RLSM**	FQI	**RFQI**	LSM	DOS	NLSM	**RLSM**	FQI	**RFQI**
5	9.81 (0.05)	10.06 (0.09)	9.95 (0.09)	**9.49 (0.08)**	10.11 (0.07)	**10.09 (0.12)**	32s	40s	29s	**24s**	25s	**23s**
10	13.33 (0.05)	13.70 (0.11)	13.26 (0.14)	**12.98 (0.08)**	13.75 (0.06)	**13.74 (0.07)**	48s	42s	32s	**25s**	32s	**25s**
50	21.18 (0.04)	21.88 (0.06)	20.15 (0.34)	**21.09 (0.03)**	22.05 (0.06)	**22.02 (0.06)**	8m37s	59s	44s	**38s**	6m43s	**39s**
100	22.73 (0.07)	25.38 (0.11)	22.75 (0.43)	**24.66 (0.07)**	25.43 (0.06)	**25.56 (0.06)**	38m44s	1m17s	1m 1s	**55s**	1h18m15s	**56s**
500	-	33.43 (0.10)	30.52 (0.48)	**33.19 (0.06)**	-	**34.04 (0.04)**	-	4m18s	3m15s	**3m 3s**	-	**3m 4s**
1000	-	36.64 (0.06)	34.67 (0.29)	**36.87 (0.05)**	-	**37.68 (0.07)**	-	8m14s	5m55s	**5m47s**	-	**6m 0s**
2000	-	39.20 (0.08)	38.37 (0.38)	**40.66 (0.06)**	-	**41.47 (0.06)**	-	16m11s	11m15s	**10m51s**	-	**10m41s**

Table: Max call option on Heston for different number of stocks d. RFQI achieves the highest prices while being the fastest and having considerably less trainable parameters.

LSM: Least Squares Monte Carlo (Longstaff and Schwartz, 2001)
DOS: Deep Optimal Stopping (Becker et al., 2019)
NLSM: Neural Least Squares Monte Carlo (Lapeyre and Lelong, 2019)
RLSM: Randomized Least Squares Monte Carlo
FQI: Fitted Q-Iteration (Tsitsiklis and Van Roy, 2001)
RFQI: Randomized Fitted Q-Iteration

Geometric Put and Basket Call

		price						duration					
payoff	d	LSM	DOS	NLSM	**RLSM**	FQI	**RFQI**	LSM	DOS	NLSM	**RLSM**	FQI	**RFQI**
GeometricPut	1	7.08 (0.11)	6.97 (0.08)	7.05 (0.11)	**7.06 (0.06)**	6.92 (0.13)	**6.96 (0.10)**	8s	20s	9s	**5s**	6s	**6s**
	5	3.35 (0.04)	3.33 (0.03)	3.32 (0.05)	**3.29 (0.05)**	3.35 (0.05)	**3.34 (0.03)**	15s	22s	10s	**6s**	8s	**7s**
	10	2.38 (0.04)	2.37 (0.02)	2.35 (0.04)	**2.35 (0.03)**	2.40 (0.03)	**2.40 (0.05)**	29s	23s	10s	**7s**	13s	**7s**
	20	1.67 (0.01)	1.69 (0.03)	1.59 (0.05)	**1.64 (0.03)**	1.72 (0.02)	**1.72 (0.02)**	1m24s	23s	10s	**7s**	38s	**8s**
BasketCall	5	4.61 (0.08)	4.58 (0.06)	4.44 (0.07)	**4.61 (0.05)**	4.62 (0.04)	**4.65 (0.03)**	15s	22s	10s	**6s**	8s	**7s**
	10	3.58 (0.03)	3.60 (0.05)	3.38 (0.05)	**3.61 (0.05)**	3.61 (0.06)	**3.63 (0.03)**	30s	23s	10s	**6s**	21s	**7s**
	50	2.05 (0.03)	2.35 (0.02)	1.66 (0.07)	**2.00 (0.02)**	2.36 (0.02)	**2.37 (0.02)**	7m59s	27s	12s	**7s**	5m40s	**8s**
	100	1.18 (0.01)	2.11 (0.02)	1.24 (0.04)	**1.78 (0.01)**	2.10 (0.02)	**2.12 (0.02)**	40m34s	31s	14s	**8s**	1h17m19s	**9s**
	500	-	1.92 (0.01)	0.86 (0.02)	**1.71 (0.01)**	-	**1.95 (0.01)**	-	1m33s	30s	**15s**	-	**15s**
	1000	-	1.91 (0.01)	0.80 (0.02)	**1.77 (0.00)**	-	**1.95 (0.01)**	-	2m56s	48s	**22s**	-	**22s**
	2000	-	1.85 (0.00)	0.77 (0.01)	**1.84 (0.00)**	-	**1.96 (0.00)**	-	5m17s	1m22s	**39s**	-	**37s**

Table: Geometric put and basket call options on Black Scholes for different number of stocks d. RFQI achieves the highest prices while being the fastest and having considerably less trainable parameters.

LSM: Least Squares Monte Carlo (Longstaff and Schwartz, 2001)
DOS: Deep Optimal Stopping (Becker et al., 2019)
NLSM: Neural Least Squares Monte Carlo (Lapeyre and Lelong, 2019)
RLSM: Randomized Least Squares Monte Carlo
FQI: Fitted Q-Iteration (Tsitsiklis and Van Roy, 2001)
RFQI: Randomized Fitted Q-Iteration

Increasing the number of exercise dates

		price					duration				
d	N	LSM	DOS	NLSM	**RLSM**	**RFQI**	LSM	DOS	NLSM	**RLSM**	**RFQI**
10	50	34.54 (0.10)	36.06 (0.16)	35.95 (0.17)	**33.13 (0.21)**	**35.98 (0.14)**	2m45s	1m50s	1m 8s	**33s**	**37s**
	100	34.36 (0.10)	36.11 (0.14)	35.97 (0.20)	**32.75 (0.25)**	**35.83 (0.15)**	5m23s	3m55s	2m20s	**1m 7s**	**1m13s**
50	50	53.14 (0.08)	55.87 (0.14)	55.21 (0.21)	**52.29 (0.08)**	**55.64 (0.13)**	44m28s	2m 5s	1m15s	**37s**	**44s**
	100	52.77 (0.11)	56.08 (0.14)	55.60 (0.13)	**51.81 (0.09)**	**55.66 (0.13)**	1h30m27s	4m35s	2m34s	**1m14s**	**1m29s**
100	50	-	63.81 (0.14)	62.54 (0.22)	**60.42 (0.12)**	**63.58 (0.13)**	-	2m53s	1m25s	**42s**	**48s**
	100	-	64.10 (0.12)	63.22 (0.12)	**59.92 (0.07)**	**63.58 (0.07)**	-	5m57s	2m50s	**1m25s**	**1m37s**
500	50	-	80.86 (0.07)	76.45 (0.62)	**78.51 (0.07)**	**81.11 (0.07)**	-	8m50s	2m47s	**1m16s**	**1m20s**
	100	-	81.26 (0.11)	78.07 (0.42)	**78.15 (0.11)**	**81.21 (0.10)**	-	17m36s	5m37s	**2m34s**	**2m40s**

Table: Max call option on Black Scholes for different number of stocks d and higher number of exercise dates N. RFQI achieves similar prices as DOS while being the fastest and having considerably less trainable parameters.

LSM: Least Squares Monte Carlo (Longstaff and Schwartz, 2001)
DOS: Deep Optimal Stopping (Becker et al., 2019)
NLSM: Neural Least Squares Monte Carlo (Lapeyre and Lelong, 2019)
RLSM: Randomized Least Squares Monte Carlo
FQI: Fitted Q-Iteration (Tsitsiklis and Van Roy, 2001)
RFQI: Randomized Fitted Q-Iteration

Randomized Recurrent Least Squares Monte Carlo (RLSM)

$$\begin{aligned} p_N^i &= g(x_N^i) \\ p_n^i &= \underbrace{g(x_n^i)}_{\text{payoff}} \underbrace{\mathbf{1}_{\{g(x_n^i)\geq c_\theta(x_n^i)\}}}_{\text{exercise}} + \underbrace{\alpha p_{n+1}^i}_{\text{discounted future price}} \underbrace{\mathbf{1}_{\{g(x_n^i)< c_\theta(x_n^i)\}}}_{\text{continue}} \end{aligned}$$

- The continuation value is approximated by a **recurrent neural network**

$$\begin{cases} h_n &= \sigma(A_x x_n + A_h h_{n-1} + b) \\ c_{\theta_n}(h_n) &= A_n^\top h_n + b_n \end{cases}$$

- where the parameters of the hidden state (A_x, A_h, b) are randomly chosen and not optimized
- and only the parameters of the readout map $\theta_n = (A_n, b_n)$ are optimized by minimizing the loss function

$$\psi_n(\theta_n) = \sum_{i=1}^{m} \left(c_{\theta_n}(x_n^i) - \alpha p_{n+1}^i\right)^2 .$$

- We go forward for computing h_n and backward for computing p_n.

Fractional Brownian Motion

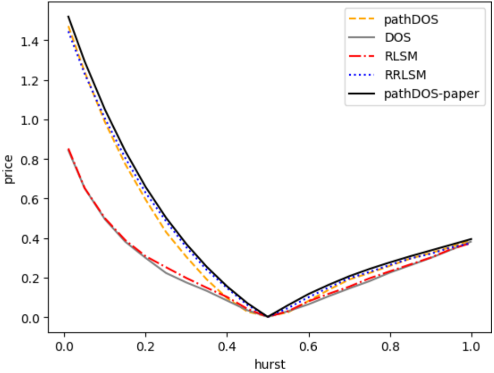

Figure: Payoff identity. Algorithms processing path information outperform.
DOS: Deep Optimal Stopping (Becker et al., 2019)
pathDOS: Deep Optimal Stopping using the entire path
pathDos-paper: values reported in the paper
RLSM: Randomized Least Squares Monte Carlo
RRLSM: Randomized Recurrent Least Squares Monte Carlo

Fractional Brownian Motion

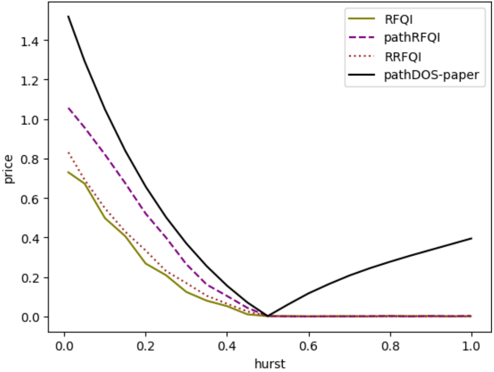

Figure: Reinforcement learning does not work well in non-Markovian case.
RFQI: Randomized Fitted Q-Iteration
pathRFQI: Randomized Fitted Q-Iteration
RRFQI: Randomized Recurrent Fitted Q-Iteration
pathDos-paper: Deep Optimal Stopping using the entire path

Fractional Brownian Motion

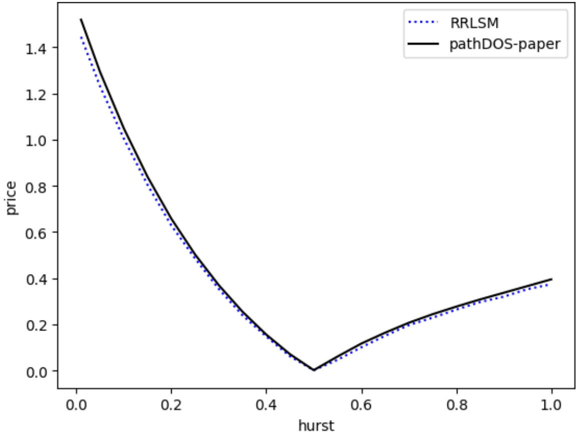

Figure: Randomized Recurrent Least Monte Carlo (RRLSM) achieves similar results as reported in deep optimal stopping, while using only 20K paths instead of 4M for training which took only 4s instead of the reported 430s

High dimensions Hurst

		price				duration			
payoff	d	DOS	pathDOS	RLSM	**RRLSM**	DOS	pathDOS	RLSM	**RRLSM**
Identity	1	0.66 (0.01)	1.24 (0.01)	0.65 (0.02)	**1.23 (0.01)**	1m15s	2m34s	3s	**4s**
Max	5	1.72 (0.04)	1.59 (0.01)	2.08 (0.01)	**2.15 (0.01)**	1m32s	12m55s	15s	**15s**
	10	1.85 (0.03)	1.68 (0.01)	2.45 (0.00)	**2.46 (0.00)**	1m47s	26m21s	22s	**24s**
Mean	5	0.29 (0.00)	0.52 (0.00)	0.28 (0.01)	**0.52 (0.01)**	1m30s	12m57s	12s	**14s**
	10	0.20 (0.01)	0.36 (0.00)	0.22 (0.02)	**0.32 (0.02)**	1m44s	25m19s	28s	**21s**

Table: Identity, maximum and mean on the fractional Brownian motion with $H = 0.05$ and different number of stocks d. RRLSM achieves high prices while being much faster than pathDOS.

DOS: Deep Optimal Stopping (Becker et al., 2019)
NLSM: Neural Least Squares Monte Carlo (Lapeyre and Lelong, 2019)
RLSM: Randomized Least Squares Monte Carlo
FQI: Fitted Q-Iteration (Tsitsiklis and Van Roy, 2001)
RFQI: Randomized Fitted Q-Iteration

Conclusion

- we introduced two simple and powerful approaches:
  - **Randomized Least Squares Monte Carlo (RLSM)**
  - **Randomized Fitted Q-Iteration (RFQI)**
- They have the advantage of the state-of-the-art: they are very simple to implement and have convergence guarantees.
- They have the advantages of neural networks; they are easily scalable to high dimensions, and there is no need to choose basis functions by hand.
- On top of that, these methods are extremely fast.
- In Markovian problems, RFQI outperforms all existing methods reconfirming that reinforcement learning methods surpass backward induction methods.
- In non-Markovian problems, our **randomized recurrent neural network algorithm (RRLSM)**, achieves similar results as the path-version of DOS, while using less training data and being much faster.
- The speed of our algorithms is very promising for applications in high dimensions and with many discretization dates.

Thank you

for your attention

Becker, S., Cheridito, P., and Jentzen, A. (2019). Deep optimal stopping. Journal of Machine Learning Research, 20:74.

Lapeyre, B. and Lelong, J. (2019). Neural network regression for bermudan option pricing.

Longstaff, F. A. and Schwartz, E. S. (2001). Valuing american options by simulation: a simple least-squares approach. Review of Financial Studies.

Tsitsiklis, J. and Van Roy, B. (2001). Regression methods for pricing complex american-style options. IEEE Transactions on Neural Networks, 12(4):694–703.